The Relationship of CS Attitudes, Perceptions of Collaboration, and Pair Programming Strategies on Upper Elementary Students' CS Learning

Jessica Vandenberg
jvanden2@ncsu.edu
North Carolina State University, USA

Arif Rachmatullah
arachma@ncsu.edu
North Carolina State University, USA

Collin Lynch
cflynch@ncsu.edu
North Carolina State University, USA

Kristy Elizabeth Boyer
keboyer@ufl.edu
University of Florida, USA

Eric Wiebe
wiebe@ncsu.edu
North Carolina State University, USA

ABSTRACT

Pair programming is a popular strategy in computer science education to teach programming to novices. In this study, we examined the effect of three different pair programming conditions on upper elementary school students' CS conceptual understanding. The three conditions were one-computer with roles (1C with roles), two computers without roles (2C no roles), and two computers with roles (2C with roles). These students were engaged in four days of computer programming activities and took the CS concept assessment, CS attitudes, and collaboration perceptions before and after the activities. We used the validated E-CSCA (Elementary Computer Science Concepts Assessment) to measure elementary students' understanding of CS concepts. We tested the relationship of different pair programming conditions on the students' CS conceptual understanding and found that different conditions impacted students' CS conceptual understanding, wherein students in 2C roles demonstrated better CS learning than the other two conditions. The results also showed no changes in students' CS attitudes and perceptions of collaboration before and after the activities. Furthermore, the results indicated no significant impact of these attitudinal factors on students' learning CS concepts in pair programming settings. Our study highlights the importance of the roles and number of computers in pair programming settings, especially for elementary students.

CCS CONCEPTS

• **Social and professional topics → K-12 education**; • **Applied computing → Collaborative learning**.

KEYWORDS

pair programming, K-5, multilevel modeling, instrument validation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ITiCSE 2021, June 26–July 1, 2021, Virtual Event, Germany
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8214-4/21/06…$15.00
https://doi.org/10.1145/3430665.3456347

ACM Reference Format:
Jessica Vandenberg, Arif Rachmatullah, Collin Lynch, Kristy Elizabeth Boyer, and Eric Wiebe. 2021. The Relationship of CS Attitudes, Perceptions of Collaboration, and Pair Programming Strategies on Upper Elementary Students' CS Learning. In *26th ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2021), June 26–July 1, 2021, Virtual Event, Germany*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3430665.3456347

1 INTRODUCTION

How students regulate their learning can profoundly influence their performance in school [30], their motivation and affect [29], and their use of strategies like help-seeking [27]. Moreover, when students effectively regulate their learning they are more likely to persist in challenging learning situations, to apply a range of strategies, and make use of feedback [4]. Most research has focused on how individual learners regulate their own learning [2, 45, 47], but a growing body of work has looked at regulation in collaborative learning contexts [eg. 15]. Research has also explored the degree to which teachers or peers [28, 34] and technology [5, 26] can influence students' academic regulation.

Teachers regularly encourage collaborative work and assign group projects across the content areas [1]; moreover, these are widely used strategies in elementary contexts (e.g., [9]). For effective collaboration to occur, all learners must successfully regulate their individual and collective learning. Regulated, collaborative learning requires negotiation of task goals, monitoring, and evaluation of diverse strategies and processes [13, 14]. Learning environments are most conducive to effective collaborative talk when students can verbally interact with each other and when they have access to strategies to consider diverse ideas [25].

The majority of research on how students collaborate in computer-supported collaborative learning (CSCL) environments are nested in university or high school settings [cf. 41]. A particularly active area of research in secondary CS education has been on collaborative (pair) programming [e.g. 24, 44]. Developmental differences, in addition to varied experiences with technology, underscore the need to examine how much younger learners interact in such environments. Relatedly, there are far fewer studies in elementary and middle school settings that explore how students collaborate in computer science-based environments in particular [e.g. 7, 36]. Moreover, studies that examine young students' social-emotional functioning and academic motivations in settings like computer

science (CS) are extremely limited. This highlights a significant gap across the fields of regulation of learning, CS education, and CSCL.

More specifically, the most significant gap across these fields might be how the technology itself mediates collaboration among elementary-aged group members. Gómez et al. [10] found that kindergarten-aged children who shared a display but who used their own mouse to make changes made vast improvements in oral language and social skills compared to the experimental group. Similarly, Zurita and Nussbaum [48] found that elementary students with lower problem-solving capacities were heightened when using technology; in particular, the children were better able to communicate, negotiate, and coordinate when they used handheld devices during collaborative math and language arts tasks. From a pragmatic standpoint, students can better express agency when they have control of their own materials; by introducing individual workstations for each learner engaged in a collaborative task, there is potential to explore differences in young students' collaboration.

The increase over the past ten years in studies aimed at examining students' collaborative work denotes educational researchers' interest in understanding the complexities involved in aligning the efforts, interests, and abilities of multiple students across diverse settings. In spite of this, there remains much to explore; namely, to what extent condition might influence elementary students' CS learning, attitudes, and their perceptions of working collaboratively.

2 BACKGROUND AND RELATED WORKS

2.1 Pair Programming in CS

Many researchers in computer science education agree generally on the benefits of pair programming. Studies involving computer science students from high school through university level indicate that pair programming enhances students' overall enjoyment of and confidence in programming [21], increases computer science (CS) competence for females and lower ability students [20], some improvement in CS self-efficacy [6], and higher grades [23]. At its core, traditional pair programming (henceforth called 1C with roles) comprises two programmers sharing one computer and alternating the driver-navigator roles after a set time or portion of the task was completed. We maintain that by incorporating an additional computer, pair programmers each can express their own agency through their input devices. However, even with two computers, students can still have assigned roles (i.e., driver and navigator) as they do in traditional (1C) pair programming. Here, we delineate this two-computer pair programming into two categories: with and without roles, 2C no Roles and 2C with Roles, respectively. All three of these conditions are explored in detail below.

2.2 Student Perceptions of Collaboration

Many would agree that collaboration is beneficial to student learning. However, it is important to explore to what extent collaboration might impact student perceptions of such activity. Lee et al. [18] conclude that when students have more experience with collaboration, their ability to collaborate and interest in collaborative activities increases. In fact, Kurucay [16] found that students who were guided to collaborate in specific ways reported more positive perceptions of collaboration than of those guided to work individually. There are noted barriers to successful collaboration, however. These include not knowing the basic conventions of collaboration, partners who asymmetrically contribute (or engage in social loafing), one partner's real or perceived competence leading to power imbalances, and working with a friend [17]. Some students simply prefer to work alone. When required to collaborate, their negative associations with group activity significantly influences their willingness to contribute to joint work [37].

2.3 Regulation of Learning

Collaborative regulation of learning is a transitional process in which an individual student acquires regulatory skills from the collaborative group member(s) [13]. These skills include appropriate planning, monitoring, and evaluation processes implemented while learning. Such regulation can be triggered in a learning activity by the student asking for assistance, by others prompting the student to engage in some strategic process, or by a technology-driven cue. One of the affordances of learning alongside others is how the processes involved in learning can be guided by, or distributed among, those in the group. An external regulating agent, such as a peer, can allow a student to focus on the task and internalize the regulatory behaviors necessary for future tasks [13]. Such collaborative regulation can help by directing the behaviors of the learner for the benefit of the group activity [11], and by providing opportunities for the learner to appropriate these regulatory skills for the future.

Roles, as typically implemented in pair programming in CS, involve each student taking on a set responsibility that is in some way complementary to his or her partner. However, designated roles are not guaranteed in pair programming. In some configurations, experienced programmers are left to naturally organize how they work together and separately [31]. The number of computers programmers use in a pair programming situation has ranged from one to four, as noted previously. For pragmatic purposes, only one or two computers will be considered for this study. Assigned roles, such as the driver and navigator used in traditional (1C) pair programming, can provide a structure within the collaborative relationship that supports regulation of learning. Roles can provide guidance for novice collaborators as to how to contribute effectively to the task and guard against either overly aggressive or passive approaches to the collaboration. Formal assignment of roles also scaffold the learning of regulatory skills that can be applied to future collaborative activity. Logistically, there is no reason why roles cannot be utilized in 2C programming configurations. However, there is a paucity of research that explores this possibility.

2.4 Current Work

The purpose of the current study was to examine the impact of different pair programming conditions on upper elementary (4th and 5th grades) students' conceptual understanding of CS, CS attitudes, and their perceptions of working in a collaborative setting. The following research questions guided the current study:

(1) How does the combination of roles and number of computers used in pair programming configurations affect the students' CS learning outcomes?
(2) Does pair programming improve elementary students' attitudes towards CS and collaboration?

3 METHODS

3.1 Participants and Context

A total of 44 elementary students (10 to 11 years old) from three classes of 5th grades were recruited to participate in the current study. They attended a single rural elementary school located in the southeastern United States. 59% of the participating students were female and 41% were male. Half of these students (50%) identified as White, 21% Latinx, 5% Native American/American Indian, 5% multiracial, 2% Asian, and the remaining 18% reported as Other.

Students were grouped into pairs by the teacher upon completion of the pre-study Collaboration survey measure such that students with similar scores (i.e., collaboration perspectives) were paired; students maintained these partnerships over the course of the study. The students participated in a series of four visual block-based coding lessons, taught by the school technology teacher who had some prior experience with teaching block-based coding. The class served as a "special" that all students attended once per week for 45 minutes, much like Music, Art, or Physical Education. Each of the three classes were randomly assigned one condition and maintained that condition for the duration of the study.

The study's curriculum covered block-based coding lessons on foundational CS concepts such as loops, conditionals, and variables. As part of the curriculum, and as appropriate by condition, students were introduced to pair programming in 1C and 2C programming configurations using the NetsBlox [3] programming environment. NetsBlox allows students on two computers to use a shared, virtual programming space where both students can interactively work on the same code.

The curriculum remained constant across the classes and was written such that the concepts increase in difficulty as the students become more familiar with the programming environment. Moreover, the lessons were designed for student-driven exploration; the pairs were given basic instruction in how to use the specific blocks in the environment, but it was expected that the students would explore and make decisions about which blocks to use, all while consulting with their partner.

3.2 Conditions

The conditions for this study involve the use of roles and computers. The resulting 2x2 matrix suggests four possible conditions: 1C with no roles, 1C with roles, 2C with no roles, and 2C with roles. However, it is not practical, nor educationally desirable, to implement 1C with no roles. This follows because in this scenario, two elementary-aged students would work on a single computer without the structures and routines set in place that would foster equitable contributions and learning that guide the logic behind the driver-navigator roles. In unstructured settings, effective learning occurs when the learner receives elaborated explanations and then constructively uses that information [42]. A 1C with no roles condition is the least likely to support this as the students are not given information nor guidance on how to talk or interact. Lewis and Shah [19] emphasize that the quality of students' participation is a measure of equity; as such 1C without roles is not likely to support equitable contributions from elementary-aged students. When students' collaborative interactions are structured, they engage in more and deeper argumentation [35]. Pragmatically, simply putting students together and telling them to or hoping they will work together does not mean it will occur; this seems even less likely to occur in a 1C with no roles condition when one student has literal control of the learning devices and no direction to share or collaborate. Moreover, giving children access to materials and assisting educators' classroom structure by not disrupting equitable participation ought to be a priority of classroom-based research. As such, this condition was not be considered.

3.2.1 1C with Roles. In traditional pair programming, the roles are driver and navigator [44]. The driver inputs all coding changes via the keyboard and mouse/trackpad, whereas the navigator checks the driver's work and makes suggestions. The programmers switch roles after a set amount of time. The students in this condition were introduced to the roles and concomitant responsibilities as well as the process: when the alarm sounds on the teacher's phone and she announces it is time to switch, the students do not change seats, but rather the driver relinquishes control of the computer and the navigator assumes driving responsibilities.

3.2.2 2C no Roles. In this condition, students work on individual, yet networked computers, so the coding changes completed on one appear on the other. Students were not given roles to guide their interactions nor talk; however, they were directed to collaboratively code, problem-solve, and debug. Moreover, they were told that both partners needed to agree on a plan and any changes made.

3.2.3 2C with Roles. In this condition, students also worked on networked individual computers. However, they were assigned roles that mirror 1C roles but leverage the physical configuration of this condition and which support their collaborative regulation. Given the subtle but important differences in roles from 1C, these roles are called by a different pair of names: reviewer-proposer. The reviewer appraised past learning at the start of the new learning task and reviewed what had been done/learned at mid-task. The proposer advanced ideas post-review for what ought to occur for the day/session and adjusted their proposal after the mid-task review. Given the brief 45-minute class, the students in this condition were directed to enact their roles twice, once at the beginning and once midway through the session.

3.3 Instruments

3.3.1 E-CSA. The CS Elementary Attitudes Survey is an 11-item Likert scale survey that queries upper elementary students on their attitudes toward and perspectives on CS, coding in particular. It is based on a validated STEM attitudes survey and has both undergone cognitive interviewing with upper elementary students to determine appropriate wording for the age [40] and has recently been psychometrically validated [39]. The items cover two psychological constructs: self-efficacy and outcome expectancy [43]. An example item is "I would like to use coding to make something new" which the students would answer from *strongly disagree* to *strongly agree* on a 5-point Likert-scale.

3.3.2 CS Conceptual Knowledge (E-CSCA). This assessment was adapted from the validated middle-grade version [32]. These items were based on Grover and Basu's [12] Focal Knowledge, Skills, and

Table 1: Psychometric Properties of E-CSCA

Item	Measure	infit MNSQ	Outfit MNSQ
Item1-Variable1	1.30	0.97	1.07
Item3-Conditional3	0.16	1.10	1.19
Item4-Loops1	-0.69	1.05	1.07
Item6-Variables3	0.53	1.17	1.22
Item7-Conditionals3	0.08	1.01	1.06
Item8-Loops2	0.10	0.95	0.92
Item9-Variables4	0.70	1.04	1.00
Item12-Variables6	0.19	0.87	0.85
Item13-Algorithms1	1.42	0.97	1.28
Item15-Algorithms3	-0.59	0.91	0.83
Item16-Algorithms4	0.52	0.98	0.98
Item19-Algorithms5	-0.98	0.71	0.73
Item20-Loops4	0.08	0.89	0.81
Item21-Loops5	-1.85	0.98	0.74
Item22-Algorithms6	-1.69	1.06	0.95
Item24-Conditionals6	0.36	0.90	0.91
Item25-Conditionals7	0.36	1.10	1.19

Abilities (FKSAs) framework and assessed students' conceptual understanding of core CS concepts (i.e., variables, loops, conditionals, and algorithms). Individual items were written mostly using block-based coding text, although others are word problems. We then tested these 25 items on 120 upper elementary school students to gain more insights into each of the items' psychometric properties. A combination of classical test theory and item response theory Rasch was used to validate this assessment before using it in the current study. We found only 18 items having acceptable infit and outfit MNSQ values, a cutoff range of 0.70 to 1.30 [46]. An item with infit and outfit MNSQ within the cutoff range means that the item could differentiate students based on their CS conceptual understanding levels (e.g., low and high achievers). This 18-item assessment also had acceptable reliability values [> .70 8] with person reliability .71, item reliability .88. Table 1 presents the psychometric properties of the final assessment.

3.3.3 Collaboration. The Collaboration Survey is a 5-item Likert-scale survey that queries students on their non-domain specific collaboration interests. It is based on collaboration work done with middle school students by [7]. An example item is "When more than one person works on a project, we do better" which the students would answer from *strongly disagree* to *strongly agree*.

3.4 Analysis

Multilevel modeling (MLM) was run using SAS to answer the research questions. MLM is frequently used to analyze repeated, longitudinal, and nested data and therefore is appropriate for the data we had. Moreover, unlike traditional multivariate analysis (e.g., ANOVA) that require balanced data, MLM can still be performed with unbalanced data. Thus, students who took only either pre-test or post-test would not be dropped and still included in the analysis [33]. Also, MLM can be performed with our current sample size with minimal bias estimation [22]. MLM is particularly useful to model intraindividual variability, such as in this study, because we examined changes in students' conceptual understanding of CS in three different conditions. Furthermore, we also sought the explore changes related to students' attitudes towards CS and collaboration. Our hypotheses about the changes in CS conceptual understanding, CS attitudes, and perceptions of collaboration (Level 1) and their association with the three conditions (Level 2) were tested using the following equations:

Level 1 (Time):
CS Conceptual Understanding$_i t$ = β_{0it} + β_{1it} (Time) + β_{2it} (CS Attitudes) + β_{3it} (Collaboration) + β_{4it} (Time*CS Attitudes) + β_{5it} (Time*Collaboration) + r_{it}

Level 2 (Student):
β_{0i} (M CS Conceptual Understanding) = γ_{00} + γ_{01} (Condition) + u_{01}
β_{1i} (Time) = γ_{10} + γ_{11} (Condition)
β_{2i} (CS Attitudes) = γ_{20}
β_{3i} (Collaboration) = γ_{30}
β_{4i} (Time*CS Attitude) = γ_{40}
β_{5i} (Time*Collaboration) = γ_{50}

Equation under Level 1 (Time) denotes the within-person relationship of CS conceptual understanding, test occasion (Time), CS attitudes, and collaboration perceptions. The intercept β_{0it} is the expected CS score for student i in the pretest, given that we coded Time as 0 (pretest) and 1 (post-test), and who had average CS attitudes and perceptions of collaboration. We grand-mean centered CS attitudes and perceptions of collaboration, so all the interpretation should be based on students with average CS attitudes and perceptions of collaboration. The first slope, β_{1it}, is called a CS learning slope specifying the CS conceptual understanding changes from pretest to post-test. The second and third slopes, β_{2it} and β_{3it}, are the expected changes in CS conceptual understanding associated with CS attitudes and perceptions of collaboration, respectively. The fourth and fifth slopes, β_{4it} and β_{5it}, are the expected changes in CS learning (CS conceptual understanding and Time) associated with changes in CS attitudes and perceptions of collaboration, respectively. The r_{it} is the residual error term that represents the variation around the mean of the CS score.

The intercept and slopes in Level 1 became the outcome variables in Level 2. The intercept γ_{00} represents the average CS score for students in the 1C with roles condition (1C with roles, coded as 0). The intercept γ_{10} represents the average relationship between CS conceptual understanding with the test occasion, and γ_{11} tests whether this relationship (i.e., CS learning) varies based on the pair programming conditions to which students are assigned. We used γ_{11} to answer the first research question. We used γ_{40} and γ_{50} to answer the second research question, given that these intercepts tested whether CS learning depended on changes in CS attitudes and perceptions of collaboration.

A null or unconditional model was performed before testing the above equation. This null model consisted of only the CS conceptual understanding score without any predictors. The null model was used to explore whether significant within- (σ^2) and between-students (τ_{00}) variability exist in the CS conceptual understanding to proceed with MLM.

4 RESULTS

4.1 Preliminary Analysis

The null or unconditional model generated the interclass correlation coefficient (ICC) [$\rho = \tau_{00} / (\tau_{00} + \sigma^2)$] that presents the amount of within- and between-student variances in the dependent variable, CS conceptual understanding. Based on the results, we found a significant ($p < .05$) within- and between-student variance in the CS conceptual understanding. The results indicated that 61% of the variability in students' CS conceptual understanding was within-student ($\sigma^2 = 167.95$, $z = 3.89$, $p < .001$) and 39% was between-student ($\tau_{00} = 109.53$, $z = 2.33$, $p = .038$). This indicated that we could run MLM in the subsequent analysis.

4.2 Multilevel Modeling Results

Table 2 presents the MLM results. It can be seen that the students' CS Learning was not significantly ($p > .05$) associated with the changes in CS attitudes and perceptions of collaboration (γ_{40} and γ_{50}, respectively). However, we found that the pair programming conditions students were assigned to were significantly ($p = .040$) associated with students' CS learning (γ_{11}). This model accounted for 34% of within-student and 7% of between-students variability in CS conceptual understanding.

Tests of significant contrast were run to decompose this interaction effect. Figure 1 visualizes these results. Looking pre to post for each condition, we found no significant increase in students' CS conceptual understanding in the 1C with roles condition (two-tailed $t = 1.37$, p = .181). In contrast, we found a statistically significant increase in CS conceptual understanding in both 2C without and with roles (two-tailed $t = 4.79$, p < .001; $t = 4.72$, $p < .001$, respectively). Moreover, we did not find significant differences ($p > .05$) in students' CS conceptual understanding before the intervention, meaning that all three groups of students had relatively similar levels of CS conceptual understanding prior to the intervention. Looking at post-test scores after the intervention, we found that students in 2C with roles had a significantly higher CS conceptual understanding than those in 2C without roles condition (one-tailed $t = 1.81$, $p = .040$) and 1C with roles (one-tailed $t = 2.95$, $p = .001$). Also, students in the 2C without roles condition had significantly higher CS conceptual understanding than students in 1C with roles after the intervention (one-tailed $t = 1.83$, $p = .039$).

5 DISCUSSION

Learning both computer science concepts and how to collaborate effectively can be a challenging prospect, especially for young children who often struggle with sharing materials and articulating their thoughts clearly to one another. We set out to explore how to scaffold upper elementary students' learning of CS concepts using three pair programming configurations: traditional 1 computer with driver-navigator roles, 2 computers without roles, and 2 computers with roles. We found that the condition students were assigned was significantly associated with learning outcomes. In particular, the students in the 2C with roles condition performed statistically significantly higher on the CS conceptual assessment.

Prior research indicates that many upper elementary students do not enjoy the 1C condition [38]. The children in that study report that they only felt like contributors when they were driving and that miscommunications occurred between driver and navigator. Tsan et al. [38] further reported that students in the 2C without roles condition felt they learned more from hands-on experience and enjoyed having independence and control over their work. We surmise that their findings are behind what we see in our results. 1C with roles students may have had intermittent engagement, likely when they were driving, and suffered from missed opportunities to learn from their partners' input and mistakes when they were navigating. 2C without roles students likely benefited from more hands-on/"driving time" because each student had a computer; however, without expectations of how and when to collaboratively talk, students lost intentional moments to talk through their thinking, their problem solving processes, and their desired next steps. To this end, it is not surprising that 2C with roles resulted in such positive outcomes. It leveraged the best attributes of the other two conditions (collaborative talk expectations supporting regulation and student agency with individual computers) to provide enriching scaffolding for students to learn.

6 CONCLUSION AND FUTURE WORK

We are one of the first to utilize synchronous virtual pair programming with elementary students and to assign roles based on foundational learning theory to make empirical conclusions about how to best scaffold young students' learning in computer science. In sum, we found that collaborative regulation matters, that it helps students' academic accomplishments, and that a simple intervention such as assigning roles in pair programming can positively affect how students perform. Our findings have readily applicable strategies for practitioners, including the simplicity of assigning roles to students. Limits on in-class devices, however, mean that not all teachers and students can benefit from our findings regarding the 2C condition. Teachers must make decisions over student technology use and the practical limitations they face in their classrooms. That noted, policy-makers at the district or state level ought to consider the long-term investment of providing sufficient devices to classrooms in order to address not only CS learning gaps but also to bolster students' collaborative regulation. Lastly, our findings have implications for theory by contributing to the foundational work of others in CS education who research collaborative work, in particular pair programming. Our work was not without limitation, however. Our sample size was rather small and we only gathered data from a single school. This small sample, resulting in low power, may contribute to the insignificant findings with CS attitudes and collaboration; results from a larger sample may show different results. Future work ought to address these limitations. Moreover, qualitative work providing triangulating evidence, such as analysis of students' collaborative discourse, would enrich the findings.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No DRL 1721160. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Table 2: Unstandardized Coefficients (and Standard Errors) of Multilevel Models of CS Conceptual Understanding (Notes: Reference group/Intercept = Pretest, average CS Attitudes, and collaboration; no asterisk $p > .05$, $^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$)

Effects	Parameter	Unconditional	Model 1
CS Conceptual Understanding, β_0			
Intercept	γ_{00}	39.19***(2.21)	31.67***(4.00)
Condition	γ_{01}		0.30 (3.10)
CS Learning slope, β_1			
Time(Post-test)	γ_{10}		5.99 (4.63)
Condition	γ_{11}		7.24* (3.37)
CS Attitudes slope, β_2			
CS Attitudes	γ_{20}		-1.74 (3.76)
Collaboration slope, β_3			
Collaboration	γ_{30}		-4.16 (0.27)
CS Learning x Attitudes slope, β_4			
Time x CS Attitudes	γ_{40}		0.65 (7.29)
CS Learning x Collaboration slope, β_5			
Time x Collaboration	γ_{50}		-1.05 (6.33)
Random Effects			
Between-student (τ_{00})		109.53* (52.66)	101.36* (45.21)
Within-person fluctuation (σ^2)		167.95*** (43.22)	110.33*** (33.36)

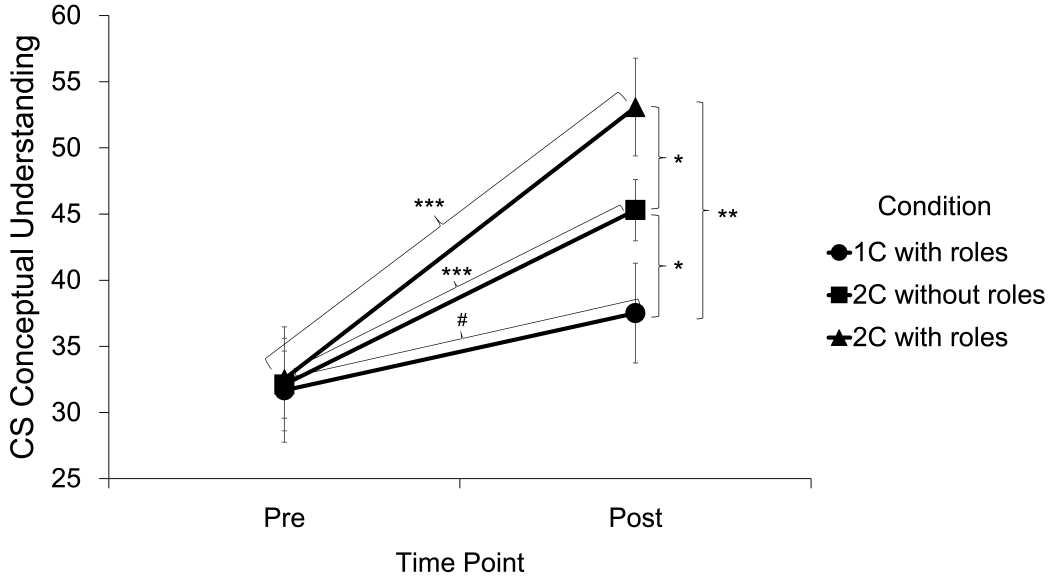

Figure 1: Students' CS Learning Based on Pair Programming Conditions(Notes: # $p > .05$, $^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$)

REFERENCES

[1] Phyllis C Blumenfeld, Ronald W Marx, Elliot Soloway, and Joseph Krajcik. 1996. Learning with peers: From small group cooperation to collaborative communities. *Educational researcher* 25, 8 (1996), 37–39.

[2] Monique Boekaerts. 1996. Self-regulated learning at the junction of cognition and motivation. *European psychologist* 1, 2 (1996), 100.

[3] Brian Broll, Akos Lédeczi, Peter Volgyesi, Janos Sallai, Miklos Maroti, Alexia Carrillo, Stephanie L Weeden-Wright, Chris Vanags, Joshua D Swartz, and Melvin Lu. 2017. A visual programming environment for learning distributed programming. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, Seattle, Washington, USA, 81–86.

[4] Deborah L Butler and Philip H Winne. 1995. Feedback and self-regulated learning: A theoretical synthesis. *Review of educational research* 65, 3 (1995), 245–281.

[5] Nada Dabbagh and Anastasia Kitsantas. 2012. Personal Learning Environments, social media, and self-regulated learning: A natural formula for connecting formal and informal learning. *The Internet and higher education* 15, 1 (2012), 3–8.

[6] K Davidson, L Larzon, and Karl Ljunggren. 2010. Self-efficacy in programming among STS students. *Retrieved August* 12 (2010), 2013.

[7] Jill Denner, Linda Werner, Shannon Campe, and Eloy Ortiz. 2014. Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education* 46, 3 (2014), 277–296.

[8] RF DeVellis. 2016. Scale development: theory and applications 4th edth ed.

[9] Robyn M Gillies and Michael Boyle. 2010. Teachers' reflections on cooperative learning: Issues of implementation. *Teaching and teacher Education* 26, 4 (2010), 933–940.

[10] Florencia Gómez, Miguel Nussbaum, Juan F Weitz, Ximena Lopez, Javiera Mena, and Alex Torres. 2013. Co-located single display collaborative learning for early childhood education. *International Journal of Computer-Supported Collaborative Learning* 8, 2 (2013), 225–244.

[11] Valeska Grau and David Whitebread. 2012. Self and social regulation of learning during collaborative activities in the classroom: The interplay of individual and group cognition. *Learning and Instruction* 22, 6 (2012), 401–412.

[12] Shuchi Grover and Satabdi Basu. 2017. Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*. ACM, Seattle, Washington, USA, 267–272.

[13] Allyson Fiona Hadwin, Sanna Järvelä, and Mariel Miller. 2011. Self-regulated, co-regulated, and socially shared regulation of learning. *Handbook of self-regulation of learning and performance* 30 (2011), 65–84.

[14] Sanna Järvelä and Hanna Järvenoja. 2011. Socially constructed self-regulated learning and motivation regulation in collaborative learning groups. *Teachers College Record* 113, 2 (2011), 350–374.

[15] Sanna Järvelä, Paul A Kirschner, Ernesto Panadero, Jonna Malmberg, Chris Phielix, Jos Jaspers, Marika Koivuniemi, and Hanna Järvenoja. 2015. Enhancing socially shared regulation in collaborative learning groups: designing for CSCL regulation tools. *Educational Technology Research and Development* 63, 1 (2015), 125–142.

[16] Murat Kurucay. 2015. *Examining the effects of learner-learner interaction on students' perceptions of collaboration, sense of community, satisfaction, perceived learning and achievement in an online undergraduate course*. Ph.D. Dissertation. Texas Tech University.

[17] Ha Le, Jeroen Janssen, and Theo Wubbels. 2018. Collaborative learning practices: teacher and student perceived obstacles to effective student collaboration. *Cambridge Journal of Education* 48, 1 (2018), 103–122.

[18] Silvia Wen-Yu Lee and Chin-Chung Tsai. 2011. Students' perceptions of collaboration, self-regulated learning, and information seeking in the context of Internet-based learning and traditional learning. *Computers in human behavior* 27, 2 (2011), 905–914.

[19] Colleen M Lewis and Niral Shah. 2015. How equity and inequity can emerge in pair programming. In *Proceedings of the eleventh annual international conference on international computing education research*. ACM, Omaha, Nebraska, USA, 41–50.

[20] Phil Maguire, Rebecca Maguire, Philip Hyland, and Patrick Marshall. 2014. Enhancing collaborative learning using paired-programming: Who benefits? *AISHE-J: The All Ireland Journal of Teaching and Learning in Higher Education* 6, 2 (2014), 1411–14125.

[21] Charlie McDowell, Linda Werner, Heather E Bullock, and Julian Fernald. 2006. Pair programming improves student retention, confidence, and program quality. *Commun. ACM* 49, 8 (2006), 90–95.

[22] Daniel M McNeish and Laura M Stapleton. 2016. The effect of small sample size on two-level model estimates: A review and illustration. *Educational Psychology Review* 28, 2 (2016), 295–314.

[23] Emilia Mendes, Lubna Basil Al-Fakhri, and Andrew Luxton-Reilly. 2005. Investigating pair-programming in a 2nd-year software development and design computer science course. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*. ACM, Monte de Caparica, Portugal, 296–300.

[24] Marcello Missiroli, Daniel Russo, and Paolo Ciancarini. 2016. Learning agile software development in high school: an investigation. In *Proceedings of the 38th International Conference on Software Engineering Companion*. ICSE, Austin, Texas, USA, 293–302.

[25] Luisa Molinari and Consuelo Mameli. 2013. Process quality of classroom discourse: Pupil participation and learning opportunities. *International Journal of Educational Research* 62 (2013), 249–258.

[26] Daniel C Moos and Roger Azevedo. 2008. Self-regulated learning with hypermedia: The role of prior domain knowledge. *Contemporary Educational Psychology* 33, 2 (2008), 270–298.

[27] Richard S Newman. 2002. How self-regulated learners cope with academic difficulty: The role of adaptive help seeking. *Theory into practice* 41, 2 (2002), 132–138.

[28] David J Nicol and Debra Macfarlane-Dick. 2006. Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in higher education* 31, 2 (2006), 199–218.

[29] Paul R Pintrich. 1999. The role of motivation in promoting and sustaining self-regulated learning. *International journal of educational research* 31, 6 (1999), 459–470.

[30] Paul R Pintrich and Elisabeth V De Groot. 1990. Motivational and self-regulated learning components of classroom academic performance. *Journal of educational psychology* 82, 1 (1990), 33.

[31] Lutz Prechelt, Ulrich Stärk, and Stephan Salinger. 2008. 7 types of cooperation episodes in Side-by-Side programming. In *Proceedings 21st Annual Meeting of the Psychology of Programming Interest Group*. PPIG, Limerick, Ireland, 148–161.

[32] Arif Rachmatullah, Bita Akram, Danielle Boulden, Bradford Mott, Kristy Boyer, James Lester, and Eric Wiebe. 2020. Development and validation of the middle grades computer science concept inventory (MG-CSCI) assessment. *EURASIA Journal of Mathematics, Science and Technology Education* 16, 5 (2020), em1841.

[33] Stephen W Raudenbush and Anthony S Bryk. 2002. *Hierarchical linear models: Applications and data analysis methods*. Vol. 1. Sage Publications, Thousand Oaks, CA.

[34] JS Rozendaal, A Minnaert, and M Boekaerts. 2005. The influence of teacher perceived administration of self-regulated learning on students' motivation and information-processing. *Learning and Instruction* 15, 2 (2005), 141–160.

[35] Oliver Scheuer, Bruce M McLaren, Maralee Harrell, and Armin Weinberger. 2011. Will structuring the collaboration of students improve their argumentation?. In *International Conference on Artificial Intelligence in Education*. Springer, AIED, Berlin, Germany, 544–546.

[36] Niral Shah, Colleen Lewis, and Roxane Caires. 2014. Analyzing equity in collaborative learning situations: A comparative case study in elementary computer science. In *ICLS 2014 Proceedings*. ACM, Boulder, Colorado, USA.

[37] Hyo-Jeong So and Thomas A Brush. 2008. Student perceptions of collaborative learning, social presence and satisfaction in a blended learning environment: Relationships and critical factors. *Computers & education* 51, 1 (2008), 318–336.

[38] Jennifer Tsan, Jessica Vandenberg, Zarifa Zakaria, Joseph B Wiggins, Alexander R Webber, Amanda Bradbury, Collin Lynch, Eric Wiebe, and Kristy Elizabeth Boyer. 2020. A Comparison of Two Pair Programming Configurations for Upper Elementary Students. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, Portland, Oregon, USA, 346–352.

[39] Jessica Vandenberg, Arif Rachmatullah, Collin Lynch, Kristy Elizabeth Boyer, and Eric Wiebe. in review. Relationship between race and gender in elementary computer science attitudes: A validation and cross-sectional study.

[40] Jessica Vandenberg, Jennifer Tsan, Danielle Boulden, Zarifa Zakaria, Collin Lynch, Kristy Elizabeth Boyer, and Eric Wiebe. 2020. Elementary students' understanding of CS terms. *ACM Transactions on Computing Education (TOCE)* 20, 3 (2020), 1–19.

[41] Freydis Vogel, Christof Wecker, Ingo Kollar, and Frank Fischer. 2017. Socio-cognitive scaffolding with computer-supported collaboration scripts: A meta-analysis. *Educational Psychology Review* 29, 3 (2017), 477–511.

[42] Noreen M Webb and Ann Mastergeorge. 2003. Promoting effective helping behavior in peer-directed groups. *International Journal of Educational Research* 39, 1-2 (2003), 73–97.

[43] Allan Wigfield and Jacquelynne S Eccles. 2000. Expectancy–value theory of achievement motivation. *Contemporary educational psychology* 25, 1 (2000), 68–81.

[44] Laurie Williams, Robert R Kessler, Ward Cunningham, and Ron Jeffries. 2000. Strengthening the case for pair programming. *IEEE software* 17, 4 (2000), 19–25.

[45] Christopher A Wolters. 2003. Regulation of motivation: Evaluating an underemphasized aspect of self-regulated learning. *Educational psychologist* 38, 4 (2003), 189–205.

[46] BD Wright and JM Linacre. 1994. Reasonable mean-square fit values. *Rasch Measurement Transactions* 30 (1994), 370.

[47] Barry J Zimmerman. 2002. Becoming a self-regulated learner: An overview. *Theory into practice* 41, 2 (2002), 64–70.

[48] Gustavo Zurita and Miguel Nussbaum. 2004. Computer supported collaborative learning using wirelessly interconnected handheld computers. *Computers & education* 42, 3 (2004), 289–314.